

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-088891

(43)Date of publication of application : 30.03.1999

(51)Int.Cl.

H04N 7/32

(21)Application number : 10-194893

(71)Applicant : NEC CORP

(22)Date of filing : 10.07.1998

(72)Inventor : NAITO YUKIHIRO
MIYAZAKI TAKASHI
KURODA ICHIRO

(30)Priority

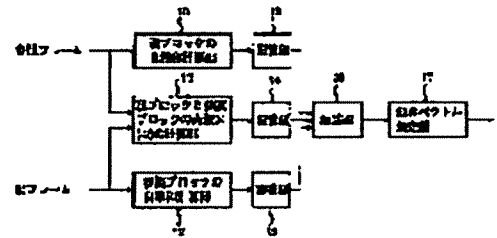
Priority number : 07195998 Priority date : 07.07.1995 Priority country : JP

(54) MOVEMENT DETECTION SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To sharply reduce operation quantity in a block matching type movement detection system for detecting the moving vector of a moving image by using a processor provided with an arithmetic logical computing unit and a product-sum computing unit.

SOLUTION: The square sum of pixel values of a current block which is calculated by a calculation part 10 is stored in a storage part 13. A calculation part 11 calculates a value obtained by multiplying an inner product between a current block and a part or all of reference blocks in a reference area by (-2) in each reference block and stores the calculated value in a storage part 14. A calculation part 12 calculates the square sum of pixel values in a part or all of reference blocks in the reference area in each reference block and stores the calculated value in a storage part 15. An addition part 16 mutually adds the square sum of the current block, the (-2) multiplied value of the inner product between the current block corresponding to a moving vector to be retrieved and the reference block, and the square sum of the reference block to find out an error power value. A moving vector determination part 17 detects a minimum value of error power and determines an optimum moving vector.



(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平11-88891

(43)公開日 平成11年(1999) 3月30日

(51)Int.Cl.⁶
H 0 4 N 7/32

識別記号

F I
H 0 4 N 7/137

Z

審査請求 有 請求項の数 2 O L (全 22 頁)

(21)出願番号 特願平10-194893
(62)分割の表示 特願平7-273032の分割
(22)出願日 平成7年(1995)10月20日
(31)優先権主張番号 特願平7-195998
(32)優先日 平7(1995)7月7日
(33)優先権主張国 日本 (J P)

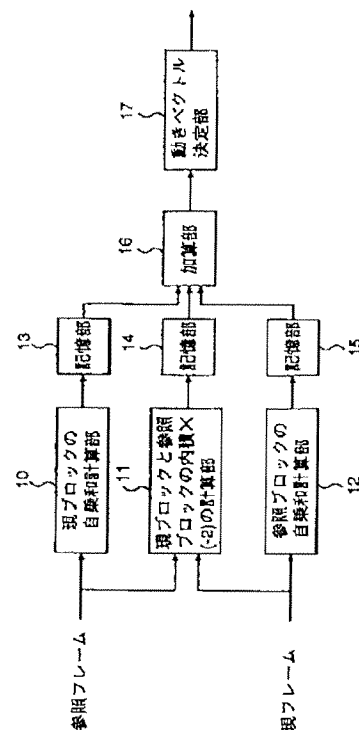
(71)出願人 000004237
日本電気株式会社
東京都港区芝五丁目7番1号
(72)発明者 内藤 幸宏
東京都港区芝五丁目7番1号 日本電気株
式会社内
(72)発明者 宮崎 孝
東京都港区芝五丁目7番1号 日本電気株
式会社内
(72)発明者 黒田 一朗
東京都港区芝五丁目7番1号 日本電気株
式会社内
(74)代理人 弁理士 京本 直樹 (外2名)

(54)【発明の名称】 動き検出方式

(57)【要約】

【課題】 算術論理演算器と積和演算器を備えたプロセッサを用いて動画像の動きベクトルを検出するブロックマッチング型動き検出方式に関し、演算量を大幅に削減することを目的とする。

【解決手段】 10にて計算された現ブロックの画素値の自乗和を13に記憶する。11にて、現ブロックと参照領域内の一部、あるいは全ての参照ブロックとの内積を-2倍した値を参照ブロック毎に計算し、14に記憶する。12にて、参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和を参照ブロック毎に計算し、15に記憶する。現ブロックの自乗和と、探索する動きベクトルに対応する現ブロックと参照ブロックの内積を-2倍した値と、参照ブロックの自乗和を16にて加算し誤差電力値とする。17にて、誤差電力の最小値を検出し、最適な動きベクトルを決定する。



【特許請求の範囲】

【請求項1】フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを検出する動き検出方式において、前記現ブロックの画素値の自乗和を計算する手段と、前記現ブロックと前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックとの内積を -2 倍した値を該参照ブロック毎に計算する内積計算手段と、前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第1の参照ブロック自乗和演算手段と、現ブロックの画素値の自乗和と、現ブロックと参照ブロックとの内積を -2 倍した値と、参照ブロックの画素値の自乗和とを加算した誤差電力値を順次計算する手段と、前記順次計算された誤差電力値のうち最小となるものを検出し、誤差電力値が最小となったときの参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力する手段とを備える動き検出方式において、前記第1の参照ブロック自乗和演算手段に代えて、前記参照フレーム内の全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第2の参照ブロック自乗和演算手段と、該参照ブロック自乗和演算結果を記憶する記憶手段を備え、複数の異なる現ブロックの動き検出において同一の参照ブロックの画素値の自乗和として、前記記憶手段に記憶された参照ブロック自乗和演算結果を再利用することを特徴とする動き検出方式。

【請求項2】フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを検出する動き検出方式において、前記現ブロックと前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックとの内積を -2 倍した値を該参照ブロック毎に計算する内積計算手段と、前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第1の参照ブロック自乗和演算手段と、現ブロックと参照ブロックとの内積を -2 倍した値と、参照ブロックの画素値の自乗和とを加算した誤差電力評価値を順次計算する手段と、前記順次計算された誤差電力評価値のうち最小となるものを検出し、誤差電力評価値が最小となったときの参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力する手段とを備える動き検出方式において、前記第1の参照ブロック自乗和演算手段に代えて、前記参照フレーム内の全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第2の参照ブロック自乗和演算手段と、該参照ブロック自乗和演算結果を記憶する記憶手段を備え、複数の異なる現ブロックの動き検出において同一の参照ブロックの画素値の自乗和として、前記記憶手段に記憶された参照ブロック自乗和演算

結果を再利用することを特徴とする動き検出方式。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、動き検出方式に関し、特に、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを検出する動き検出方式に関する。

【0002】

【従来の技術】動画像の圧縮符号化においては、動き補償と直交変換を組み合わせた動画像圧縮符号化方式が多く用いられ、動き補償において、画像の動きを検出するために、一般にブロックマッチング型動き検出方式が用いられる。従来のブロックマッチング型動き検出方式としては、例えば特開平1-295379号公報に示されたものが知られている。図13は従来の動き検出方式を説明するものであって、(a)は現フレームを示し、

(b)は参照フレームを示している。フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを決定する。動画像において連続するフレーム間では動きは小さいと考えられるために、図13のように参照フレーム中で、相関の高いブロックを探索する範囲を参照領域として設定し、この領域内の全ての参照ブロック、或は意図的に選択された参照ブロックに対して相関を計算する。意図的に参照ブロックを選択する手法としては、1993年発行の動画像符号化規格であるISO/IEC11172-2の82頁に記載されているロガリズムクサーチ法がある。ロガリズムクサーチ法は、例えば、動きベクトルが(0, 0)とその周辺 ± 4 の8個の参照ブロックを探索し、その中で最適参照ブロックを決定し、次に、決定された最適参照ブロックの周辺 ± 2 の8個の参照ブロックを探索するといった方法で探索していくものである。現フレームの全てのブロックに対して、以上のような探索処理を実行し動きベクトルを検出する。探索の際の評価量である相関は、一般に、現ブロックと参照ブロックの画素値の差分の絶対値、或は画素値の差分の自乗値をブロック全体に対して加算することによって評価される。

【0003】図14に算術論理演算器と積和演算器を有するプロセッサの例を示す。相関を評価する際に、現ブロックと参照ブロックの画素値の差分の絶対値和を用いる場合には、現ブロックと参照ブロックの画素値に対して、算術論理演算器において差分演算と絶対値演算と加算演算を実行するため、1画素あたり3サイクルの演算が必要となる。また、評価する際に、現ブロックと参照ブロックの画素値の差分の自乗和を用いる場合には、算術論理演算器で差分演算を、積和演算器で自乗和演算をするため、1画素あたり2サイクルの演算で実行することができる。図15に相関の評価に自乗和を用いたときのフローチャートを示す。以後、現ブロックと参照ブ

ックの画素値の差分の自乗和を誤差電力値と呼ぶ。ブロックの大きさを $N \times N$ 個、現ブロックの画素値を $x_{i,j}$ 、参照領域の画素値を $y_{i,j}$ とし、動きベクトルが (u, v) のときの誤差電力値は次式で示される。ただし *

$$e_{u,v} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (x_{i,j} - y_{i+u,j+v})^2 \quad (1)$$

【0005】積和演算に必要な時間を 1 サイクル、差分演算に必要な時間を 1 サイクルとすると、1 ブロック分の誤差電力の計算には、 $2 \times N \times N$ サイクル必要である。この計算を相関を計算する全ての参照ブロックに対して行うことで、1 つのブロックの動きベクトルが検出される。この手順を現フレーム内の全ての選択されたブロックに対して順次行う。

【0006】

【発明が解決しようとする課題】図 15 に示すように算術論理演算器と積和演算器を備えたプロセッサで動き検出をする場合、現ブロックと 1 つの参照ブロック間の相関の計算に 1 画素当たり 2 サイクル必要であり、これをブロック内全ての画素に対して行う必要がある。1 つの現ブロックに対して複数の参照ブロックとの相関を計算する必要があり、この処理を 1 つの現フレーム内の全てのブロックに対して実行しなければならないため、膨大な演算量となる。一方、ビデオ会議等で用いられる画像圧縮符号化には実時間処理が要求される。画像圧縮符号化処理においては、動き検出部分の演算量が大きな比重を占めており、装置規模を小さくする、或は性能の向上を図るためには、動き検出部分の演算量をできる限り削減することが要求される。そこで、本発明の目的は、算術論理演算器と積和演算器を備えたプロセッサで動き検出をする際に必要となる演算量を大幅に削減することにある。

【0007】

【課題を解決するための手段】第 1 の発明による動き検出方式は、フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを検出する動き検出方式において、前記現ブロックの画素値の自乗和を計算する手段と、前記現ブロックと前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックとの内積を -2 倍した値を該参照ブロック毎に計算する内積計算手段と、前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第 1 の参照ブロック自乗和演算手段と、現ブロックの画素値の自乗和と、現ブロックと参照ブロックとの内積を -2 倍した値と、参照ブロックの画素値の自乗和とを加算した誤差電力値を順次計算する手段と、前記順次計算された誤差電力値のうち最小となるものを検出し、誤差電力値が最小となったと

* し、 N は 1 より大きい整数であり、 $x_{0,0}$ を現ブロックの左上の画素値とする。

【0004】

【数 1】

きの参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力する手段とを備えていることを特徴とする。

【0008】第 2 の発明による動き検出方式は、フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを検出する動き検出方式において、前記現ブロックと前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックとの内積を -2 倍した値を該参照ブロック毎に計算する内積計算手段と、前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第 1 の参照ブロック自乗和演算手段と、現ブロックと参照ブロックとの内積を -2 倍した値と、参照ブロックの画素値の自乗和とを加算した誤差電力評価値を順次計算する手段と、前記順次計算された誤差電力評価値のうち最小となるものを検出し、誤差電力評価値が最小となったときの参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力する手段とを備えていることを特徴とする。

【0009】第 3 の発明による動き検出方式（請求項に係る発明）は、第 1、あるいは、第 2 の発明において、前記第 1 の参照ブロックの自乗和演算手段に代えて、前記参照フレーム内の全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第 2 の参照ブロック自乗和演算手段と、該参照ブロック自乗和演算結果を記憶する記憶手段を備え、複数の異なる現ブロックの動き検出において同一の参照ブロックの画素値の自乗和として、前記記憶手段に記憶された参照ブロック自乗和演算結果を再利用することを特徴とする。

【0010】第 4 の発明による動き検出方式は、第 1、第 2、あるいは、第 3 の発明において、前記第 1 または第 2 の参照ブロック自乗和演算手段が、参照ブロックの水平方向または垂直方向のブロック毎の自乗和からブロック全体の自乗和の計算をする手段とで構成されることを特徴とする。

【0011】第 5 の発明による動き検出方式は、第 1、第 2、第 3、あるいは、第 4 の発明において、前記内積計算手段が、現ブロックの画素値を水平方向、垂直方向共に 2 で間引いた 4 種類のブロックから加減算により計算される計 9 個の副次現ブロックを出力とする現ブロック前処理手段と、参照領域の画素値を水平方向、垂直方

向共に 2 で間引いた 4 種類の領域から加減算により計算される計 9 個の副次参照領域を出力とする参照領域前処理手段と、前記副次現ブロックと前記副次参照領域の内積を計算する 9 個の副次内積計算手段と、前記 9 個の副次内積計算手段の出力から、加減算により内積結果を計算する後処理手段とで構成されることを特徴とする。

【0012】第 6 の発明による動き検出方式は、第 5 の発明において、前記副次内積計算手段として、第 5 の発明に記載の内積計算手段を再帰的に用いることを特徴とする。

【0013】第 7 の発明による動き検出方式は、フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出する際に、前記参照フレームの参照領域内で、基準参照ブロックと基準参照ブロックの周辺に位置する参照ブロックを候補として最適な参照ブロックを決定し、これを基準参照ブロックとする操作を繰り返して動きベクトルを検出する動き検出方式において、前記基準参照ブロックの画素値の自乗和と前記周辺の参照ブロックの画素値の自乗和との差分をそれぞれ計算する手段と、現ブロックと基準参照ブロックの内積を -2 倍した値を計算する手段と、現ブロックと参照ブロックの内積を 2 倍した値を計算する手段と、前記計算結果と、前記記憶手段に記憶された基準参照ブロックの画素値の自乗和と当該参照ブロックの画素値の自乗和との差分値と、現ブロックと基準参照ブロックの内積を -2 倍した値を加算する手段と、各探索参照ブロックについて順次計算された前記加算結果から値が最小となるものを検出し、前記最小*

$$e_{u,v} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{i,j}^2 - 2 \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{i,j} y_{i+u,j+v} + \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} y_{i+u,j+v}^2 \quad (2)$$

【0017】式 (1) をこのように展開することにより、各項の演算量を減少させたり、各項のデータを他に有効に用いたりすることができる。以下、この点について詳述する。

【0018】上式の第 1 項は、現ブロックの画素値の自乗和である。この値は参照ブロックの位置を表す (u, v) に係わらず一定であるので、1 つの現ブロックの動きベクトルを検出する際に 1 回だけ計算すればよい。第 2 項は、現ブロックと参照ブロックの内積結果を -2 倍したものである。この計算には 1 ブロックあたり $N \times N + 1$ サイクル必要である。この計算は探索する参照ブロック全てに対して行う必要があり、式 (1) の場合の約半分の演算量である。第 3 項は、参照ブロックの画素値の自乗和であり、1 ブロックあたり $N \times N$ サイクル必要

* 値が負の値のときは前記最小値となったときの参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力し、前記最小値が負の値でないときは基準参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力する手段を備えていることを特徴とする。

【0014】第 8 の発明による動き検出方式は、フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを検出する動き検出方式において、前記現ブロックの画素値の自乗和を計算し、前記現ブロックと前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックとの内積を -2 倍した値を該参照ブロック毎に計算し、前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算し、現ブロックの画素値の自乗和と、現ブロックと参照ブロックとの内積を -2 倍した値と、参照ブロックの画素値の自乗和とを加算した誤差電力値を順次計算し、前記順次計算された誤差電力値のうち最小となるものを検出し、誤差電力値が最小となったときの参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力することを特徴とする。

【0015】

【作用】第 1 の発明による動き検出方式の作用を説明する。式 (1) を次式のように展開する。

【0016】

【数 2】

である。この計算も第 2 項と同様に探索する参照ブロック全てについて計算する必要がある。その結果、第 2 項と第 3 項の計算で式 (1) と等しいサイクル数が必要となり、これに第 1 項の計算が必要となるので、演算量の削減にはならない。

【0019】しかしながら、探索する各参照ブロックについて第 3 項の自乗和の計算を行う場合、重複する画素が多く存在する。図 8 にこの様子を示す。簡単化のため、ブロックサイズを 4×4 、動きベクトルの探索範囲を -2 ~ +2 としている。この場合、参照領域の大きさは 8×8 となる。図 8 には、例として動きベクトルが (0, 0) の場合と、(1, -1) の場合の参照ブロックの様子が示されている。この場合、ブロック内の 16 個の画素のうち 9 個の画素が 2 つの参照ブロック間で重

複している。つまり、1つの動きベクトルを検出する際に、探索する全ての参照ブロックの画素の自乗和を予め計算する必要があるが、その際に重複する画素の自乗演算を省くことにより、第3項の計算に必要なサイクル数を大幅に減らすことができる。これにより、1つの現ブロックあたりに必要となる計算量を減らすことができ結果として、計算に必要なサイクル数を減らすことができる。

【0020】また式(2)の第2項は、2次元畳み込み演算であり、FFT(高速フーリエ変換)や第5の発明 10 で示した方法を用いることにより、その計算量を減らすことが可能である。

【0021】さらに、式(2)の第1項は、現ブロックの画素値の自乗和であって、CCITTの勧告H.261等の画像の圧縮符号化においては、動き検出後に実行されるINTRA/INTERの予測モード判定時に用いることができる値である。

【0022】第2の発明による動き検出方式の作用を説明する。第1の発明と同様に、式(1)を式(2)のように展開して計算する。誤差電力値である式(2)を、 20 探索する全ての(u, v)について計算し、最小値を示す(u, v)を検出する必要がある。しかし、式(2)の第1項は(u, v)に関わらず一定値であるので、式(2)の大小比較をする際には不必要な項である。第1の発明では第1項を1回だけ計算し、この値を再利用していたが、第2の発明では第1項を計算せずに、第2項と第3項の和を誤差電力評価値として最小値検出の際の*

$$f_{u,v} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} y_{i+u,j+v}^2 \quad (3)$$

【0027】例えば、探索する動きベクトルを

【0028】

【数4】

$$u, v = -\frac{M}{2} + 1 \sim \frac{M}{2}$$

【0029】とし、この参照領域内の全ての参照ブロックを探索する場合を考える。ただし、Mは偶数とする。 40 この場合、式(3)を

【0030】

【数5】

$$u, v = -\frac{M}{2} + 1 \sim \frac{M}{2}$$

*比較に用いる。このようにすることにより、第1の発明の作用である、第1項の値をINTRA/INTERの予測モード判定時に再利用できるという特徴と、誤差電力の最小値を求めることができるという特徴は消滅するが、第1項の計算をする必要がないため、演算量をさらに削減することができる。

【0023】第3の発明による動き検出方式の作用を説明する。第3の発明では現フレーム内の各ブロックに対応する参照領域にも重複する部分があることを利用する。つまり、異なる参照領域中の参照ブロックの間にも重複する画素が存在する。例えば図9にこのような例を示す。隣あったブロックをブロック1とブロック2とする。これらの参照領域は図9に示すように重複している。例えば、ブロック1に関する動きベクトルが(2, -2)の参照ブロックと、ブロック2に関する動きベクトルが(-2, -2)の参照ブロックは全く同一になる。よって、1フレーム分の動き検出処理をする際に必要な第3項の値を、予め計算しておくことで、第3項の計算に必要なサイクル数を更に減らすことができる。

【0024】第4の発明による動き検出方式の作用を説明する。第1、第2、第3の発明同様に式(1)を式(2)のように展開して動き検出を行う。本発明は、式(2)の第3項を参照領域の水平方向と垂直方向で分離して計算することにより演算量を削減するものである。

【0025】式(2)の第3項を $f_{u,v}$ とする。

【0026】

【数3】

【0031】について計算することになる。

【0032】第1の発明の作用でも述べたように、参照領域中の参照ブロック間には重複する画素が多く存在する。この重複する画素の自乗計算を省くことにより、第3項に必要な演算量を大幅に削減することができる。

【0033】本発明では、この第3項の計算を水平方向、垂直方向に分離して行う。例えば水平方向、垂直方向の順に計算する場合について説明する。まず、参照ブロックの水平方向の自乗和を以下のように計算する。

【0034】

【数6】

$$f'_{k,l} = \sum_{i=0}^{N-1} y_{i+k,l}^2,$$

$$k = -\frac{M}{2} + 1, \dots, \frac{M}{2}, \quad l = -\frac{M}{2} + 1, \dots, N + \frac{M}{2} - 1$$

(4)

【0035】次に、これらの結果を垂直方向に加算する 10* 【0036】
 ことによって、第3項を求める。つまり、 * 【数7】

$$f_{u,v} = \sum_{j=0}^{N-1} f'_{u,j+v},$$

$$u, v = -\frac{M}{2} + 1, \dots, \frac{M}{2}$$

(5)

【0037】ここで、式(4)と式(5)は次のように 20* 【0038】
 巡回的に高速に計算できる。 ※ 【数8】

$$f'_{k,l} = f'_{k-1,l} + y_{k+N-1,l}^2 - y_{k-1,l}^2$$

(6)

$$f_{u,v} = f_{u,v-1} + f'_{u,v+N-1} - f'_{u,v-1}$$

(7)

【0039】この方法によれば、第3項の必要演算量を
 $N^2 \times M^2$ から $(N+M-1) \times (N+2M) + M \times$
 $(N+2M)$ に削減できる。例えば、 $N=16$, $M=3$
 2とすると、演算量は262, 144サイクルから6,
 320サイクルに削減されることになる。なお、式
 (2)の第3項を水平方向と垂直方向に分離して計算す
 る本発明は、Mが偶数、奇数によらずに適用できるもの
 である。

★【0040】第5の発明による動き検出方式の作用を説
 明する。第1、第2、第3、第4の発明同様に式(1)
 を式(2)のように展開して動き検出を行う。この際、
 式(2)の第2項の計算を以下のように行う。
 【0041】 $c_{u,v}$ を以下のように定義する。
 【0042】
 【数9】

$$c_{u,v} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{i,j} y_{i+u,j+v}$$

(8)

【0043】この $c_{u,v}$ を
 【0044】
 【数10】

$$u, v = -\frac{M}{2} + 1 \sim \frac{M}{2}$$

【0045】について計算し、-2倍することが、動き
 ベクトルの検出範囲を

【0046】

【数11】

$$u, v = -\frac{M}{2} + 1 \sim \frac{M}{2}$$

【0047】としたときの参照領域中の全ての参照ブロ
 ックに関する式(2)の第2項を計算することになる。
 ただし、Mは偶数とする。

【0048】

【外1】

$$\hat{x}_{i,j}, \hat{y}_{i,j}$$

【0050】

* * 【数12】

$$\hat{x}_{i,j} = [x_{i,j}, x_{i+2,j}, \dots, x_{i+N-2,j}] \quad (9)$$

$$\hat{y}_{i,j} = [y_{i,j}, y_{i+2,j}, \dots, y_{i+N-2,j}] \quad (10)$$

【0051】これらを用いて、 $X_{i,j}$, $Y_{i,j}$ を以下 ※【0052】
のように定義する。 ※ 【数13】

$$X_{i,j} = [\hat{x}_{i,j}, \hat{x}_{i,j+2}, \dots, \hat{x}_{i,j+N-2}]^T \quad (11)$$

$$Y_{i,j} = [\hat{y}_{i,j}, \hat{y}_{i,j+2}, \dots, \hat{y}_{i,j+N-2}] \quad (12)$$

【0053】 $X_{i,j}$, $Y_{i,j}$ は、それぞれ $N^2 / 4$ ★【0054】これらベクトルを用いると式(8)の $c_{u,v}$, $c_{u+1,v}$, $c_{u,v+1}$, $c_{u+1,v+1}$ は以下のよ
個の要素を持つベクトルである。各要素は、それぞれ現
ブロック、参照領域の (i, j) 位置の $N \times N$ のブロッ
クの画素値を水平方向、垂直方向共に2で間引いたもの
【0055】
である。 ★20 【数14】

$$C_{u,v} = \begin{bmatrix} c_{u,v} \\ c_{u+1,v} \\ c_{u,v+1} \\ c_{u+1,v+1} \end{bmatrix} = \begin{bmatrix} Y_{u,v} & Y_{u+1,v} & Y_{u,v+1} & Y_{u+1,v+1} \\ Y_{u+1,v} & Y_{u+2,v} & Y_{u+1,v+1} & Y_{u+2,v+1} \\ Y_{u,v+1} & Y_{u+1,v+1} & Y_{u,v+2} & Y_{u+1,v+2} \\ Y_{u+1,v+1} & Y_{u+2,v+1} & Y_{u+1,v+2} & Y_{u+2,v+2} \end{bmatrix} \begin{bmatrix} X_{0,0} \\ X_{1,0} \\ X_{0,1} \\ X_{1,1} \end{bmatrix} \quad (13)$$

【0056】式(13)中には16個のYとXの内積演算が存在するが、次式の変換式を用いるとこの内積演算の個数を削減することができる。

【0057】

【数15】

$$\begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} (A+B)X - B(X-Y) \\ (B+C)Y + B(X-Y) \end{bmatrix} \quad (14)$$

【0058】式(13)中の4箇所に対して式(14) * 【0059】
 の変形を施し、まとめると $C_{u,v}$ は以下のように表さ 【数16】
 れる。 *

$$\begin{aligned} C_{u,v} &= \begin{bmatrix} (Y_{u,v} + Y_{u+1,v})X_{0,0} - Y_{u+1,v}(X_{0,0} - X_{1,0}) \\ (Y_{u+1,v} + Y_{u+2,v})X_{1,0} + Y_{u+1,v}(X_{0,0} - X_{1,0}) \\ (Y_{u,v+1} + Y_{u+1,v+1})X_{0,0} - Y_{u+1,v+1}(X_{0,0} - X_{1,0}) \\ (Y_{u+1,v+1} + Y_{u+2,v+1})X_{1,0} + Y_{u+1,v+1}(X_{0,0} - X_{1,0}) \end{bmatrix} \\ &+ \begin{bmatrix} (Y_{u,v+1} + Y_{u+1,v+1})X_{0,1} - Y_{u+1,v+1}(X_{0,1} - X_{1,1}) \\ (Y_{u+1,v+1} + Y_{u+2,v+1})X_{1,1} + Y_{u+1,v+1}(X_{0,1} - X_{1,1}) \\ (Y_{u,v+2} + Y_{u+1,v+2})X_{0,1} - Y_{u+1,v+2}(X_{0,1} - X_{1,1}) \\ (Y_{u+1,v+2} + Y_{u+2,v+2})X_{1,1} + Y_{u+1,v+2}(X_{0,1} - X_{1,1}) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Y_{u,v} + Y_{u+1,v} & Y_{u,v+1} + Y_{u+1,v+1} \\ Y_{u,v+1} + Y_{u+1,v+1} & Y_{u,v+2} + Y_{u+1,v+2} \end{bmatrix} \begin{bmatrix} X_{0,0} \\ X_{0,1} \end{bmatrix} \\ &+ \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} Y_{u+1,v} + Y_{u+2,v} & Y_{u+1,v+1} + Y_{u+2,v+1} \\ Y_{u+1,v+1} + Y_{u+2,v+1} & Y_{u+1,v+2} + Y_{u+2,v+2} \end{bmatrix} \begin{bmatrix} X_{1,0} \\ X_{1,1} \end{bmatrix} \\ &+ \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} Y_{u+1,v} & Y_{u+1,v+1} \\ Y_{u+1,v+1} & Y_{u+1,v+2} \end{bmatrix} \begin{bmatrix} X_{0,0} - X_{1,0} \\ X_{0,1} - X_{1,1} \end{bmatrix} \end{aligned} \quad (15)$$

【0060】次に、式(15)中の3箇所に対して式 【0061】
 (14)の変形を施し、まとめると以下ようになる。 50 【数17】

$$\begin{aligned}
C_{u,v} = & \begin{bmatrix} (Y_{u,v} + Y_{u+1,v} + Y_{u,v+1} + Y_{u+1,v+1})X_{0,0} \\ (Y_{u+1,v} + Y_{u+2,v} + Y_{u+1,v+1} + Y_{u+2,v+1})X_{1,0} \\ (Y_{u,v+1} + Y_{u+1,v+1} + Y_{u,v+2} + Y_{u+1,v+2})X_{0,1} \\ (Y_{u+1,v+1} + Y_{u+2,v+1} + Y_{u+1,v+2} + Y_{u+2,v+2})X_{1,1} \end{bmatrix} \\
+ & \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} (Y_{u,v+1} + Y_{u+1,v+1})(X_{0,0} - X_{0,1}) \\ (Y_{u+1,v+1} + Y_{u+2,v+1})(X_{1,0} - X_{1,1}) \end{bmatrix} \\
+ & \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} (Y_{u+1,v} + Y_{u+1,v+1})(X_{0,0} - X_{1,0}) \\ (Y_{u+1,v+1} + Y_{u+1,v+2})(X_{0,1} - X_{1,1}) \end{bmatrix} \\
+ & \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} Y_{u+1,v+1}(X_{0,0} - X_{1,0} - X_{0,1} + X_{1,1}) \quad (16)
\end{aligned}$$

【0062】 $C_{u,v}$ 、 $C_{u+1,v}$ 、 $C_{u,v+1}$ 、 $C_{u+1,v+1}$ を同時に計算し、さらに式(14)を利用して変形することにより、式(13)の16個の内積演算が、式(16)の9個の副次内積演算に削減される。ブロックサイズが $N \times N$ 、探索点数が $M \times M$ の場合、式(8)、あるいは式(13)を直接計算すると $N^2 \times M^2$ サイクル必要である。式(16)のように変形することによって、副次内積演算の演算量は合計で $9/16 \times (N^2 \times M^2)$ サイクルに削減される。これに加

えて、式(16)中の $X_{0,0} - X_{0,1}$ 、 $X_{1,0} - X_{1,1}$ 、 $X_{0,0} - X_{1,0}$ 、 $X_{0,1} - X_{1,1}$ 、 $X_{0,0} - X_{1,0} - X_{0,1} + X_{1,1}$ の現ブロックの前処理に $5N^2/4$ サイクル、 $Y_{u,v} + Y_{u+1,v} + Y_{u,v+1} + Y_{u+1,v+1}$ 、 $Y_{u+1,v} + Y_{u+2,v} + Y_{u+1,v+1} + Y_{u+2,v+1}$ 、 $Y_{u,v+1} + Y_{u+1,v+1} + Y_{u,v+2} + Y_{u+1,v+2}$ 、 $Y_{u+1,v+1} + Y_{u+2,v+1} + Y_{u+1,v+2} + Y_{u+2,v+2}$ の参照領域の前処理に $2(N+M-2) \times (N+M-1)$

+ (N+M-2)² サイクル、各副次内積演算結果から式(16)に従いC_{u,v}を計算する後処理に10M²/4サイクル必要となる。N=16、M=32の場合、副次内積演算部が147×10³サイクルに対し、前処理、後処理部は合計で9.32×10³サイクルであり、これらのオーバーヘッドは十分小さいものとなる。

【0063】図10に本発明に基づく式(2)の第3項の内積演算部を示す。式(8)の内積演算が、式(16)のように9個の副次内積演算に分割される。現ブロックの前処理にて、現ブロックの画素値から式(16)中のXで表される副次現ブロックを計算する。分割された副次内積演算部に対し、参照領域の前処理にて参照領域画素値から生成された式(16)中のYで表される副次参照領域値が入力される。各副次内積演算部の出力値から式(16)に従って、後処理にてC_{u,v}が生成される。

【0064】第6の発明による動き検出方式の作用を説明する。第6の発明では第5の発明において分割された副次内積演算を、式(16)を用いて、さらに、再帰的に分割する。例えば、第5の発明において分割された9個の副次内積演算全てに対して、さらに分割を行うと、計81個の副次内積演算となる。9個の副次内積演算に分割した場合の、副次内積演算部の演算量は、9/16×
e_{u0,v0} - e_{u0+1,v0-1}

$$= -2 \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{i,j} y_{u0+i,v0+j} + 2 \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{i,j} y_{u0+i+1,v0+j-1} + \left\{ \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} y_{u0+i,v0+j}^2 - \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} y_{u0+i+1,v0+j-1}^2 \right\} \quad (17)$$

【0067】となる。上式の第1項は現ブロックと基準参照ブロックの内積であり、1回だけ計算する必要がある。第2項は現ブロックと参照ブロックとの内積であり、これに関しては探索する全ての参照ブロックについて計算する。第3項は、基準参照ブロックの画素の自乗和と探索する参照ブロックの画素の自乗和の差分である。

【0068】基準参照ブロックと周辺の参照ブロックとの間には共通の画素が存在し、差分を取ることからこの共通部分の計算は不要となる。ブロックサイズが4×4のときの様子を図11に示す。基準参照ブロックと基準参照ブロックからみた動きベクトルが(1, -1)の参照ブロックとの間には9個の共通画素が存在するので、

* + (N² × M²) サイクルであったが、81個に分割した場合、演算量は、(9/16)² × (N² × M²) サイクルに削減される。

【0065】第7の発明による動き検出方式の作用を説明する。本発明は、ログリズミックサーチ法に見られるような、ある基準となる参照ブロックの周辺の参照ブロックを探索する場合の演算量を削減できるものである。以後、この基準となる参照ブロックを基準参照ブロックと呼ぶ。例えば、基準参照ブロック(u0, v0)の周辺の参照ブロックである(u0+1, v0)、(u0+1, v0+1)、(u0, v0+1)、(u0-1, v0+1)、(u0-1, v0)、(u0-1, v0-1)、(u0, v0-1)、(u0+1, v0-1)との相関を計算する場合を考える。この場合、相関を計算する参照ブロックは8個で、基準参照ブロックを含めた9個の参照ブロックから最適なブロックを検出する。この場合、(u0, v0)に関する式(2)と他の8個の周辺ベクトルに関する式(2)の中から最小値を検出する必要がある。本発明では、基準ベクトルである(u0, v0)に関する式(2)と周辺ベクトルに関する式(2)との差分を用いて、最小値を示すベクトルを検出する。例えばe_{u0,v0}とe_{u0+1,v0-1}の差分は、

【0066】

【数18】

この部分の自乗値を計算する必要はない。動きベクトルを水平方向、垂直方向共に±1の範囲で探索する場合、図11のように、4点の画素について自乗値を計算する必要がなくなる。この例ではブロックサイズが4×4であるが、ブロックサイズが大きくなれば、自乗値を計算する必要のない画素はさらに増加する。例えばブロックサイズが16×16で基準参照ブロックの周囲±1を探索する場合18×18点中自乗値を計算する必要のない点は14×14点となる。

【0069】このようにして、基準参照ブロックとその周辺の参照ブロックの中から最適な参照ブロックを少ない演算量で検出できる。検出した動きベクトルを(U, V)とすれば、この動きベクトルに対応する参照ブロッ

クと現ブロックの差分の自乗和 $e_{u,v}$ は式 (17) から計算することができる。

【0070】

【発明の実施の形態】以下、本発明の実施例について図面を参照しながら説明する。図1は第1、第3の発明の実施例を示すブロック図である。現ブロックの自乗和計算部10にて、現ブロックの画素の自乗和、つまり、式

(2)の第1項を計算し、記憶部13に記憶する。内積計算部11にて、現ブロックと参照ブロックの内積値を-2倍したもの、つまり、式(2)の第2項を全ての参

照ブロック、あるいは、一部の参照ブロックについて計算し、記憶部14に記憶する。参照ブロックの自乗和計算部12にて、参照ブロック画素の自乗和、つまり、式(2)の第3項を全ての参照ブロック、あるいは、一部の参照ブロックについて計算し、記憶部15に記憶する。加算部16において、記憶部13に記憶された現ブ

ロックの画素値の自乗和と記憶部14、記憶部15に記憶された計算結果の中で、探索中の動きベクトルに対応する結果を加算し式(2)の誤差電力とする。加算部16で計算された誤差電力を動きベクトル決定部17で比較し、最小の誤差電力を示す動きベクトルを出力する。

【0071】内積計算部11と参照ブロックの自乗和計算部12では探索する一部、あるいは全ての動きベクトルに対応する値を計算する。一部の動きベクトルに対応する値を計算する場合には、全て計算する場合に比べて、演算量削減の効果は小さくなるが、記憶部14、記憶部15の記憶容量を削減することができる。

【0072】図2は、第2、第3の発明の実施例を示すブロック図である。図1と異なり、現ブロックの自乗和の計算を行わない。このようにすることにより、第1の

発明の作用である、第1項の値をINTRA/INTERの予測モード判定時に再利用できるという特徴と、誤差電力の最小値を求めることができるという特徴は消滅するが、第1項の計算をする必要がないため、演算量をさらに削減することができる。

【0073】以下に各発明の詳細な実施例を示す。図3は第1の発明に基づき、1つの現ブロックの動きベクトルを検出する際の処理過程を示すものである。ステップ30で、最初に探索する動きベクトルを設定する。また、ステップ36で用いる最小値クリアしておく。ステップ31で、現ブロックの画素値の自乗和である式

(2)の第1項を計算し記憶する。ステップ32で、現ブロックと、参照領域内の一部、あるいは全ての参照ブロックとの内積を-2倍した値をそれぞれ計算し記憶する。ステップ33で、参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和をそれぞれ計算し記憶する。ステップ34～ステップ39で各動きベクトルを探索する。ステップ34で、現在候補となっている動きベクトルに対応する式(2)の第2、第3項が計算済みかどうかを確認し、計算済みであれば、ステップ35で

これらの値と式(2)の第1項を加算し誤差電力を求める。計算済みでなければ、ステップ32に戻り、必要な値を計算する。ステップ32、33で必要な値を全て計算した場合には、ステップ34の確認は不要となる。ステップ36では、計算した誤差電力が探索してきた動きベクトルの中で最小値であったかを判定する。最小値であった場合にはステップ37で最小値を更新し、このときの動きベクトルを記憶しておく。ステップ38では、探索する動きベクトルについて全て探索が終了したかを判定し、未了であればステップ39にて次の動きベクトルを設定してステップ34に戻る。終了していれば、ステップ40で、誤差電力の最小値と最小値に対応する動きベクトルを出力する。

【0074】図4は第2の発明による動き検出方式の処理過程を示す図である。図3の処理過程と違い、式

(2)の第1項の計算をせず、ステップ45で、式(2)の第2項と第3項を加算した値を誤差電力評価値として、ステップ46で比較している。そのために、ステップ50では最終結果として動きベクトルのみを出力し、誤差電力の最小値は出力しない。

【0075】第3の発明は、第1、第2の発明における式(2)の第3項の計算を1個の動きベクトルの検出に必要な値ではなく、現フレーム全てのブロックの動きベクトルの検出に必要な第3項の値の一部、あるいは全ての場合について計算する。図5は第3の発明に基づき、現フレーム全ての動きベクトルを検出する際の処理過程を示す図である。本図は一実施例として第2の発明と第3の発明を組み合わせた場合を示している。ステップ60で、1フレーム分の動きベクトルの検出に必要な式(2)の第3項を全て計算し記憶する。ステップ61で、あるブロックの動きベクトルを探索する際の最初に探索する動きベクトルを設定し、ステップ64で用いる誤差電力評価値の最小値をクリアしておく。ステップ62で、設定された動きベクトルの検出に必要な式(2)の第2項の全てを計算し記憶する。ステップ63で、計算済みの式(2)の第2項、第3項を加算し誤差電力評価値を求める。ステップ64で、この誤差電力評価値が最小値であるかを判定する。最小値であれば、ステップ65で最小値を更新し、このときの動きベクトルを保存しておく。ステップ66で、全ての動きベクトル候補の探索が終了したかを判定し、終了していなければステップ67で次の動きベクトル候補を設定しステップ63に戻る。終了したならば、ステップ68で、検出された動きベクトルを出力する。ステップ69で、1フレーム分の動きベクトルを全て検出したかを判定し、検出したならば終了し、いていなければ、ステップ61に戻り次のブロックの動きベクトルの検出を行う。

【0076】第4の発明の一実施例として、式(2)の第3項の演算を水平方向、垂直方向の順に分離して計算する方式を示す。簡単化のためにブロックサイズを4×

4、参照領域の大きさを 8×8 とし、便宜上、参照領域の画素を $y_{-2,-2} \sim y_{5,5}$ とする。この場合、参照領域中に含まれる参照ブロックの数は水平方向 5 個、垂直方向 5 個の計 25 個となる。この参照ブロックを全て探索する、このため、動きベクトルの範囲を $-2 \sim 2$ とする。

$$f'_{-2,l} = \sum_{i=-2}^1 y_{i,l}^2, \quad l = -2, -1, \dots, 5 \quad (18)$$

【0079】により計算する。次に垂直方向の演算として、式(5)の $f_{-2,-2} \sim f_{2,2}$ を式(7)の再帰式を用いて計算し記憶する。ただし、初期値 $f_{u,-2}$ は、

$$f_{u,-2} = \sum_{j=-2}^1 f'_{u,j}, \quad u = -2, -1, \dots, 2$$

(19)

【0081】により計算する。このようにして計算した式(2)の第3項を用いて、第1、第2、あるいは第3の発明による動き検出方式で動き検出を行う。例えば、ブロックのサイズを 16×16 、フレームのサイズを 176×144 、動き検出の範囲を $-7 \sim 7$ としこの範囲内の参照ブロックを全て探索する場合を考える。積和演算、算術演算共に 1 サイクルで実行するプロセッサ型動き検出装置の場合、式(1)を計算して評価する従来法では、1 フレーム分の動きを検出するのに、 11.4×10^6 サイクル必要である。本第4の発明と第1の発明を用いた手法により同様の動き検出を実行する場合、1 フレーム分の動きの検出が、 6.01×10^6 サイクルで実現可能である。これは従来法の 52.7% の演算量に相当する。本第4の発明と第1と第3の発明を組み合わせる手法により同様の動き検出を実行する場合、1 フレーム分の動きの検出が、 5.85×10^6 サイクル

$$\begin{aligned} X_{0,0} - X_{0,1} &= [X_{0,0}, X_{2,0}, X_{0,2}, X_{2,2}]^T - [X_{0,1}, \\ &X_{2,1}, X_{0,3}, X_{2,3}]^T = [X_{0,0} - X_{0,1}, X_{2,0} - X_{2,1}, \\ &X_{0,2} - X_{0,3}, X_{2,2} - X_{2,3}]^T \quad (20) \end{aligned}$$

次に、式(16)中の参照領域の前処理を行う。上記第5番目の内積演算に関する参照領域前処理値である $Y_{u,v}$

$$\begin{aligned} (u, v) \text{ が、} (-2, -2), (0, -2), (-2, 0), (0, 0) \text{ につ} \\ \text{いて上式を計算する必要があるが、それぞれの } (u, v) \text{ について } Y_{u,v} - \\ Y_{u+1,v+1} \text{ を計算すると以下になる。} (u, v) = (-2, -2) \text{ のと} \\ \text{きは、} Y_{-2,-1} - Y_{-1,-1} = [y_{-2,-1}, y_{0,-1}, y_{-2,1}, y_{0,1}] - [\\ y_{-1,-1}, y_{1,-1}, y_{-1,1}, y_{1,1}] = [y_{-2,-1} - y_{-1,-1}, y_{0,-1} - y_{1,-1}, \\ y_{-2,1} - y_{-1,1}, y_{0,1} - y_{1,1}] \quad (21) \end{aligned}$$

$$\begin{aligned} (u, v) = (0, -2) \text{ のときは、} \\ Y_{0,-1} - Y_{1,-1} = [y_{0,-1}, y_{2,-1}, y_{0,1}, y_{2,1}] - [y_{1,-1}, y_{3,-1}, \\ y_{1,1}, y_{3,1}] = [y_{0,-1} - y_{1,-1}, y_{2,-1} - y_{3,-1}, y_{0,1} - y_{1,1}, \\ y_{2,1} - y_{3,1}] \quad (22) \end{aligned}$$

(u, v) = (-2, 0) のときは、

【0077】まず水平方向の演算として、式(4)の $f'_{-2,-2} \sim f'_{2,5}$ を式(6)の再帰式を用いて計算し記憶する。ただし、初期値 $f'_{-2,1}$ は、

【0078】
【数19】

※【0080】
【数20】

★で実現可能である。これは従来法の 51.3% の演算量に相当する。

【0082】第5の発明の第1の実施例として、式(2)の第2項の演算方法を示す。簡単化のためにブロックサイズを 4×4 とし、探索する動きベクトルを $-2 \sim 1$ までの 4×4 個とする。この場合の参照領域は、 7×7 となる。

【0083】式(2)の第2項の計算は、この場合、式(16)の $C_{-2,-2}, C_{0,-2}, C_{-2,0}, C_{0,0}$ を計算する。それぞれの計算は9個の副次内積演算からなるが、ここでは、式(16)中の第5番目の副次内積演算である $(Y_{u,v+1} + Y_{u+1,v+1} - (X_{0,0} - X_{0,1}))$ について説明する。まず、式(16)中の現ブロックの前処理を行う。上記第5番目の内積演算に関する副次現ブロックである $X_{0,0} - X_{0,1}$ は次のようになる。

【0084】

$$\begin{aligned} \star Y_{u,v+1} - Y_{u+1,v+1} \text{ の計算について説明する。} \end{aligned}$$

23

24

$$\begin{aligned} Y_{-2,1} - Y_{-1,1} &= [y_{-2,1}, y_{0,1}, y_{-2,3}, y_{0,3}] - [y_{-1,1}, y_{1,1}, \\ y_{-1,3}, y_{1,3}] &= [y_{-2,1} - y_{-1,1}, y_{0,1} - y_{1,1}, y_{-2,3} - y_{-1,3}, \\ y_{0,3} - y_{1,3}] \end{aligned} \quad (23)$$

(u, v) = (0, 0) のときは、

$$\begin{aligned} Y_{0,1} - Y_{1,1} &= [y_{0,1}, y_{2,1}, y_{0,3}, y_{2,3}] - [y_{1,1}, y_{3,1}, y_{1,3}, y_{3,3}] \\ &= [y_{0,1} - y_{1,1}, y_{2,1} - y_{3,1}, y_{0,3} - y_{1,3}, y_{2,3} - y_{3,3}] \end{aligned} \quad (24)$$

となる。以上の式(21)から式(24)には16回の差分演算が含まれる。しかしながら、これらには重複する演算が含まれているので、 $y_{-2,1} - y_{-1,1}$ 、 $y_{0,1} - y_{1,1}$ 、 $y_{2,1} - y_{3,1}$ 、 $y_{-2,3} - y_{-1,3}$ 、 $y_{0,3} - y_{1,3}$ 、 $y_{2,3} - y_{3,3}$ で表される9回の差分演算のみを行い、副次参照領域を計算する。この副次参照領域の一部分を取り出すことで、式(21)から式(24)の値を生成することができる。

【0085】このようにして、計算された前処理結果をもとに、 $(Y_{u,v-1} + Y_{u+1,v+1}) (X_{0,0} - X_{0,1})$ を、(u, v) が、(-2, -2)、(0, -

* 2)、(-2, 0)、(0, 0) について計算する。

【0086】以上のような計算を式(16)中の9個の副次内積演算について行い、最終結果 $C_{u,v}$ を後処理で式(16)に従って計算する。

【0087】第5の発明の第2の実施例を示す。上記第1の実施例は式(16)に基づいて、式(2)の第2項の計算を行った。式(16)は、式(13)を式(14)を用いて変形したものである。本実施例では、式(14)の代わりに、

【0088】

【数21】

$$\begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} (A - B)X + B(X + Y) \\ (C - B)Y + B(X + Y) \end{bmatrix} \quad (25)$$

【0089】を用いて式(13)を変形した以下の式、

【数22】

【0090】

$$\begin{aligned}
C_{u,v} = & \begin{bmatrix} (Y_{u,v} - Y_{u+1,v} - Y_{u,v+1} + Y_{u+1,v+1})X_{0,0} \\ (-Y_{u+1,v} + Y_{u+2,v} + Y_{u+1,v+1} - Y_{u+2,v+1})X_{1,0} \\ (-Y_{u,v+1} + Y_{u+1,v+1} + Y_{u,v+2} - Y_{u+1,v+2})X_{0,1} \\ (Y_{u+1,v+1} - Y_{u+2,v+1} - Y_{u+1,v+2} + Y_{u+2,v+2})X_{1,1} \end{bmatrix} \\
& + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} (Y_{u,v+1} - Y_{u+1,v+1})(X_{0,0} + X_{0,1}) \\ (-Y_{u+1,v+1} + Y_{u+2,v+1})(X_{1,0} + X_{1,1}) \end{bmatrix} \\
& + \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} (Y_{u+1,v} - Y_{u+1,v+1})(X_{0,0} + X_{1,0}) \\ (-Y_{u+1,v+1} + Y_{u+1,v+2})(X_{0,1} + X_{1,1}) \end{bmatrix} \\
& + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} Y_{u+1,v+1}(X_{0,0} + X_{1,0} + X_{0,1} + X_{1,1}) \quad (26)
\end{aligned}$$

【0091】に基づき式(2)の第2項の計算を行う。
上記の計算は第1の実施例と同様に行うことができ、演算量も第1の実施例と等しい。

【0092】第6の発明の実施例として、第5の発明の第1の実施例で説明した副次内積演算、 $(Y_{u,v+1} + Y_{u+1,v+1})(X_{0,0} - X_{0,1})$ を更に分割する例

を示す。この式を $c'_{u,v}$ と置く。この $c'_{u,v}$ を (u, v) が $(-2, -2)$ 、 $(0, -2)$ 、 $(-2, 0)$ 、 $(0, 0)$ について計算する必要があった。

【0093】

【数23】

$$c'_{u,v}$$

$$\begin{aligned}
 &= (Y'_{u,v+1} + Y'_{u+1,v+1})(X_{0,0} - X_{0,1}) \\
 &= \sum_{i=0}^1 \sum_{j=0}^1 (x_{2i,2j} - x_{2i,2j+1})(y_{2i+u,2j+v+1} + y_{2i+u+1,2j+v+1})
 \end{aligned}
 \tag{27}$$

【0094】ここで、 $x'_{i,j}$ 、 $y'_{i,j}$ 、 u' 、 v' を以下のように定義する。

$$\begin{aligned}
 x'_{i,j} &= (x_{2i,2j} - x_{2i,2j+1}) \\
 y'_{i,j} &= (y_{2i,2j+1} - y_{2i+1,2j+1}) \\
 u' &= u/2 \\
 v' &= v/2
 \end{aligned}
 \tag{28}$$

31) これらを用いると、式(27)は次のように表せる。

【0096】 ※ ※ 【数24】

$$c'_{2u',2v'} = \sum_{i=0}^1 \sum_{j=0}^1 x'_{i,j} y'_{i+u',j+v'} \tag{32}$$

【0097】この式を u' 、 $v' = -1 \sim 0$ について計算することになる。この式は式(8)と同様の形式なので、第5の発明と同様の方法で式(16)のように分割し、演算量を削減することができる。

【0098】図6は第7の発明による動き検出方式の処理過程を示す図である。第7の発明による動き検出方式は、ある基準参照ブロックの周辺の参照ブロックを探索する場合に適用できるものである。ステップ70で、周辺の参照ブロックについて式(17)を計算する際に必要となる第3項を全て計算し記憶する。この計算方法については後で詳しく説明する。ステップ71で、式(17)の第1項を積和演算器にて計算し記憶する。この式(17)の第1項は現ブロックと基準参照ブロックとの内積に-2をかけたものであり、式(17)の第2項が現ブロックと参照ブロックとの内積を2倍したものであることから、以前の探索において、この項が計算済みの場合はこの計算を簡略化することができる。ステップ72で最初に探索する動きベクトル、つまり参照ブロックを設定する。またステップ74で用いる最小値をクリアしておく。次に、ステップ73～ステップ77で各参照ブロックを探索する。ステップ73では、現ブロックと参照ブロックとの内積を2倍したものである式(17)の第2項を積和演算器にて計算し、既に計算済みの第1項及び、対応する第3項と算術論理演算器にて加算する。ステップ74で、ステップ73の結果が探索してきた参照ブロック中で最小値であるかを判定する。ただ

し、ステップ73の結果が正の値であったときには、この参照ブロックよりも基準参照ブロックの方が適していることを意味するので最小値とは判断しない。最小値であった場合にはステップ75で最小値を更新し、このときの動きベクトルを記憶しておく。ステップ76では、探索する参照ブロックについて全て探索が終了したかを判定し、未了であればステップ77で次の動きベクトルを設定してステップ73に戻る。終了していれば、ステップ78で、最小値に対応する動きベクトルを出力し、対応する参照ブロックと現ブロックとの差分の自乗和を式(17)から計算して出力する。また、計算済みの現ブロックと対応する参照ブロックの内積値を2倍したものも出力する。これにより、この参照ブロックを基準ブロックとして、さらに周辺を探索する際に、式(17)の第1項の計算を簡略化することができる。ステップ70の計算についてさらに詳しく説明する。例として、ブロックサイズが 4×4 で動きベクトルを水平方向、垂直方向共に ± 1 の範囲で探索する場合を考える。この場合、図12のように、自乗値の計算が必要な画素が存在する。これを図7のようにP1～P16、Q1～Q8の領域に分割して、各領域内の画素値の自乗和をまず積和演算器にて計算する。この例では領域P1内には画素は1個しか存在しないので、この部分の自乗和は一つの画素の自乗和になる。例えば、動きベクトルを ± 4 の範囲で探索する場合には、領域P1は 4×4 のブロックになる。これら自乗和を組み合わせることにより、式(1

7) の第 3 項を計算する。例えば、図 1 2 のように基準参照ブロックからみて動きベクトルが (1, -1) の場合*

$$(Q1 + Q8 + Q7 + Q6 + Q5) - (P3 + P4 + P5 + P6 + P7)$$

(33)

のように算術論理演算器にて計算する。例えば、ブロックのサイズを 16×16 、フレームのサイズが 176×144 であり、動き検出の範囲を $-7 \sim 7$ としこの範囲内の参照ブロックをログリズミックサーチ法を用いて、まず動きベクトルが ± 4 、次に ± 2 、次に ± 1 のように探索する場合を考える。積和演算、算術演算共に 1 サイクルで実行するプロセッサの場合、式 (1) を計算して評価する従来法では、1 フレーム分の動きを検出するのに、 1.27×10^6 サイクル必要である。本発明を用いた手法で同様の動き検出を実行する場合、1 フレーム分の動きを検出するのに、 0.753×10^6 サイクルで実現可能である。これは従来法の 59.4% の演算量に相当する。

【0099】

【発明の効果】以上に説明したように本発明によれば、動画像の動きを検出する際に、式 (1) を展開して計算するようにしたので、各項の演算量を大幅に削減したり、各項のデータを他に有効に用いたりすることができるといふ効果が得られる。

【図面の簡単な説明】

【図 1】第 1、第 3 の発明の実施例を示すブロック図。

【図 2】第 2、第 3 の発明の実施例を示すブロック図。

【図 3】第 1 の発明の実施例を示すフローチャート。 ※

* 合の式 (17) の第 3 項は、

※ 【図 4】第 2 の発明の実施例を示すフローチャート。

【図 5】第 3 の発明の実施例を示すフローチャート。

【図 6】第 7 の発明の実施例を示すフローチャート。

【図 7】第 7 の発明において参照ブロックの画素値の自乗和の計算方法を説明する図。

【図 8】第 1 の発明の作用を説明する図。

【図 9】第 3 の発明の作用を説明する図。

【図 10】第 5 の発明の作用を説明するブロック図。

【図 11】第 7 の発明の作用を説明する図。

【図 12】第 7 の発明の作用を説明する図。

【図 13】動き検出方式を説明する図。

【図 14】積和演算器を有するプロセッサの例。

【図 15】従来法のフローチャート。

【符号の説明】

10 現ブロックの自乗和計算部

11 現ブロックと参照ブロックの内積計算部

12 参照ブロックの自乗和計算部

13 現ブロックの自乗和の記憶部

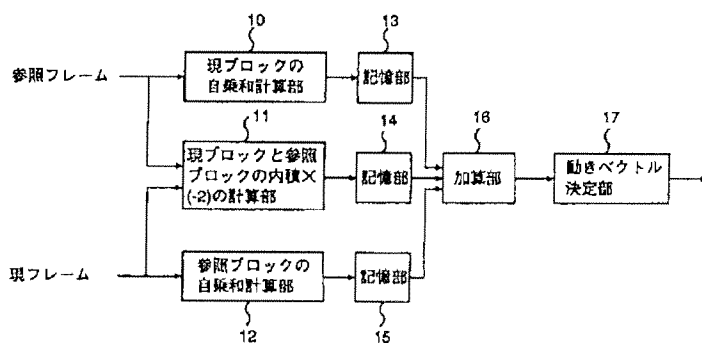
14 現ブロックと参照ブロックの内積の記憶部

15 参照ブロックの自乗和の記憶部

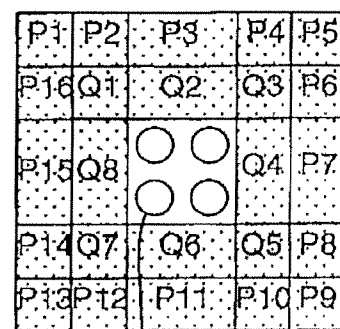
16 加算部

17 動きベクトル決定部

【図 1】

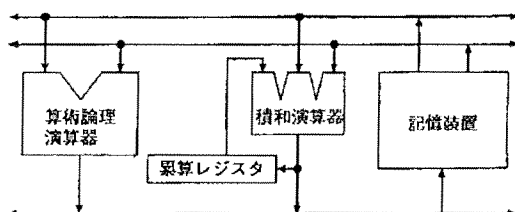


【図 7】

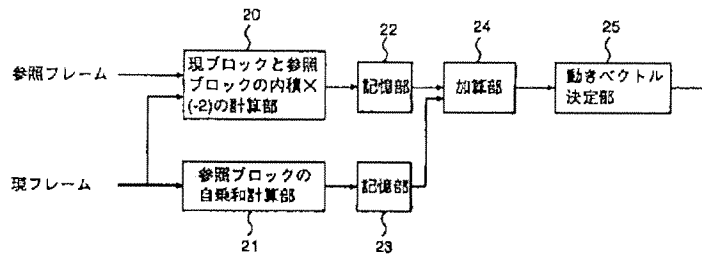


自乗計算が不必要な画素

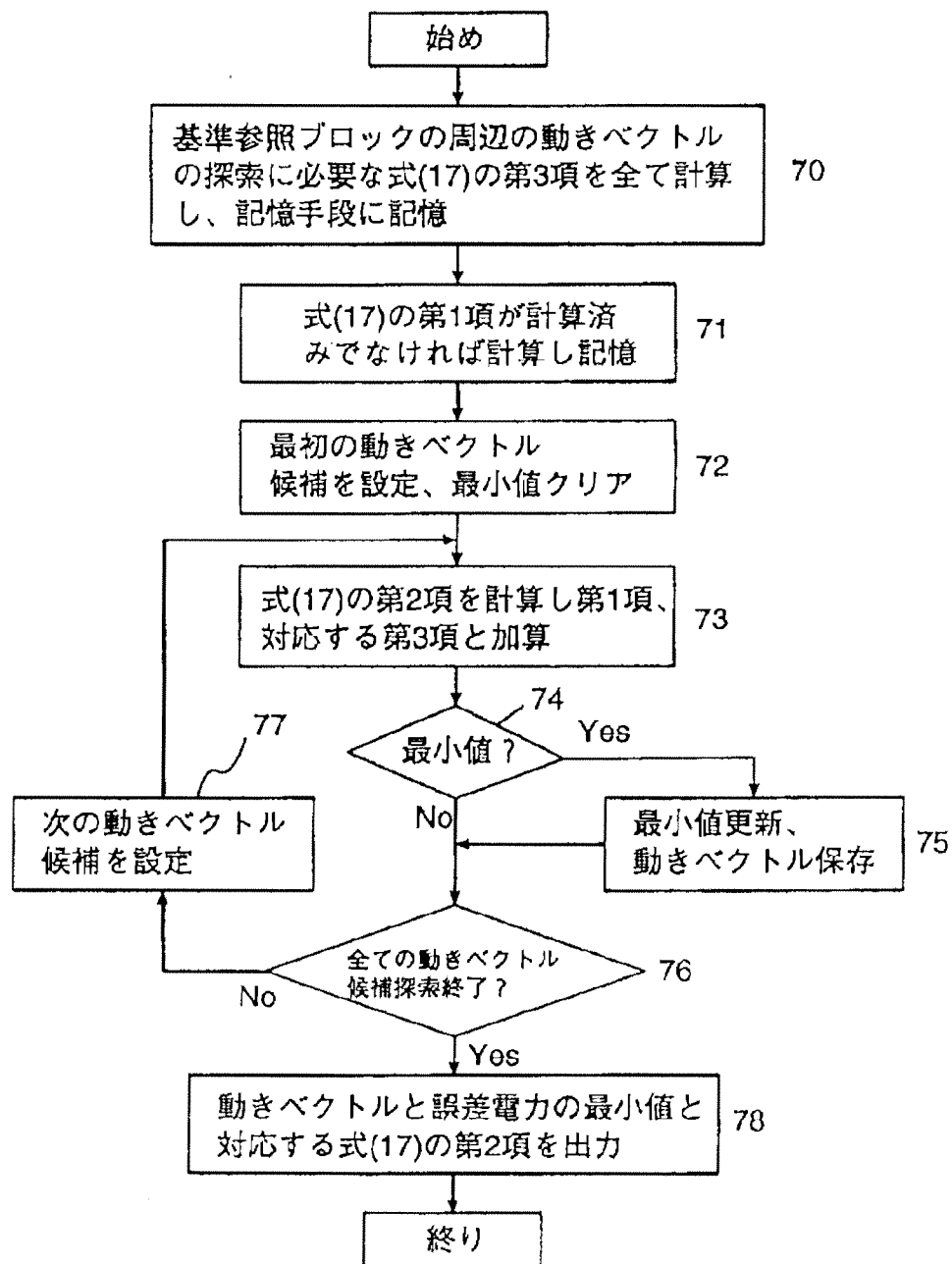
【図 14】



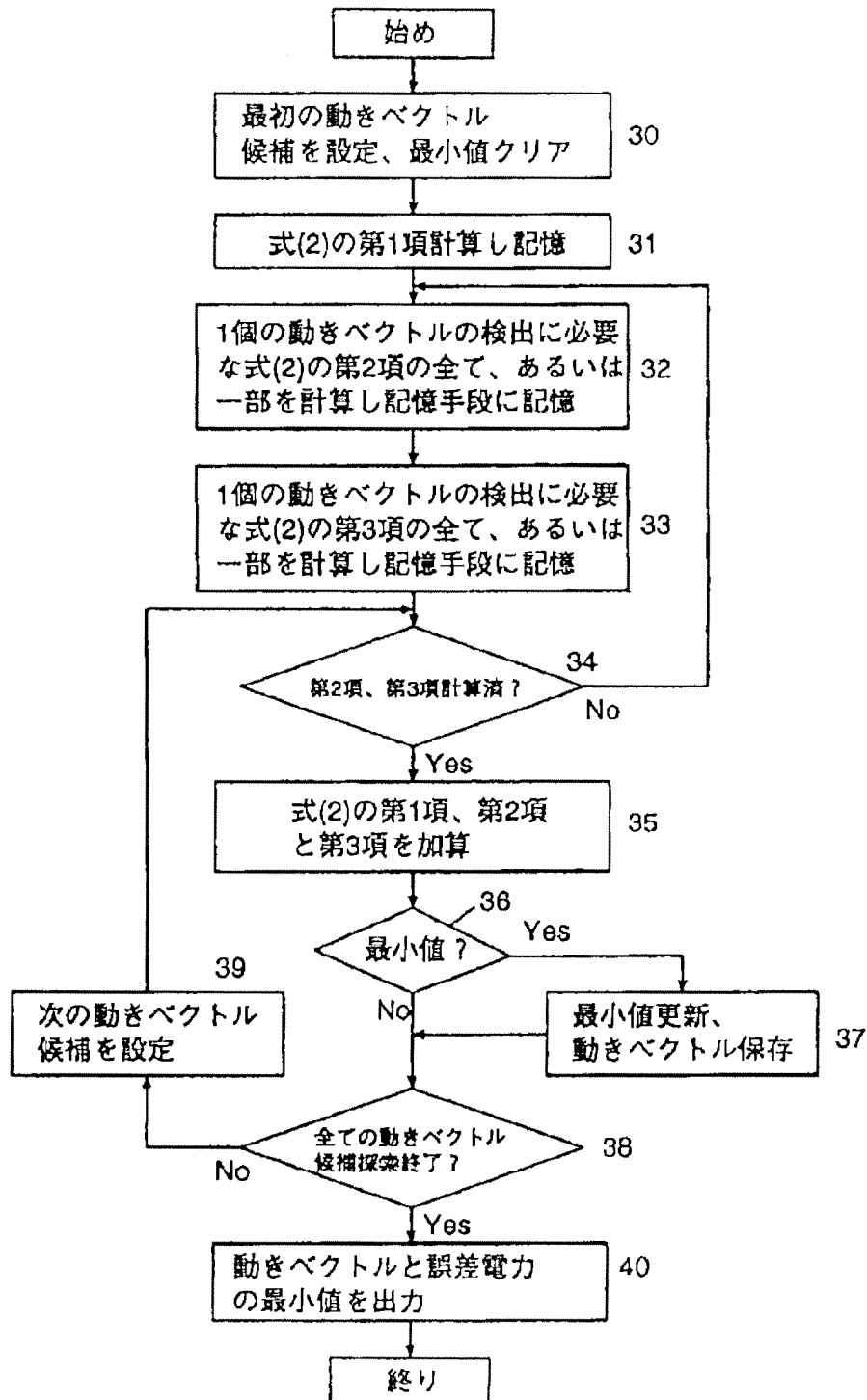
【図2】



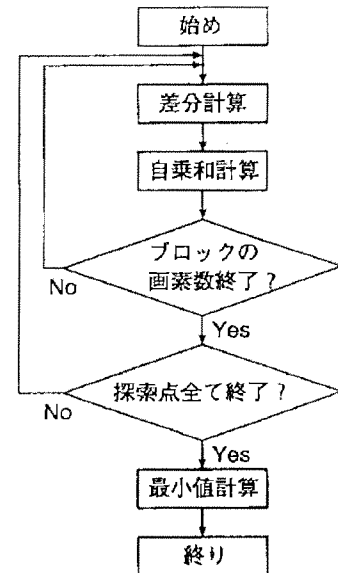
【図6】



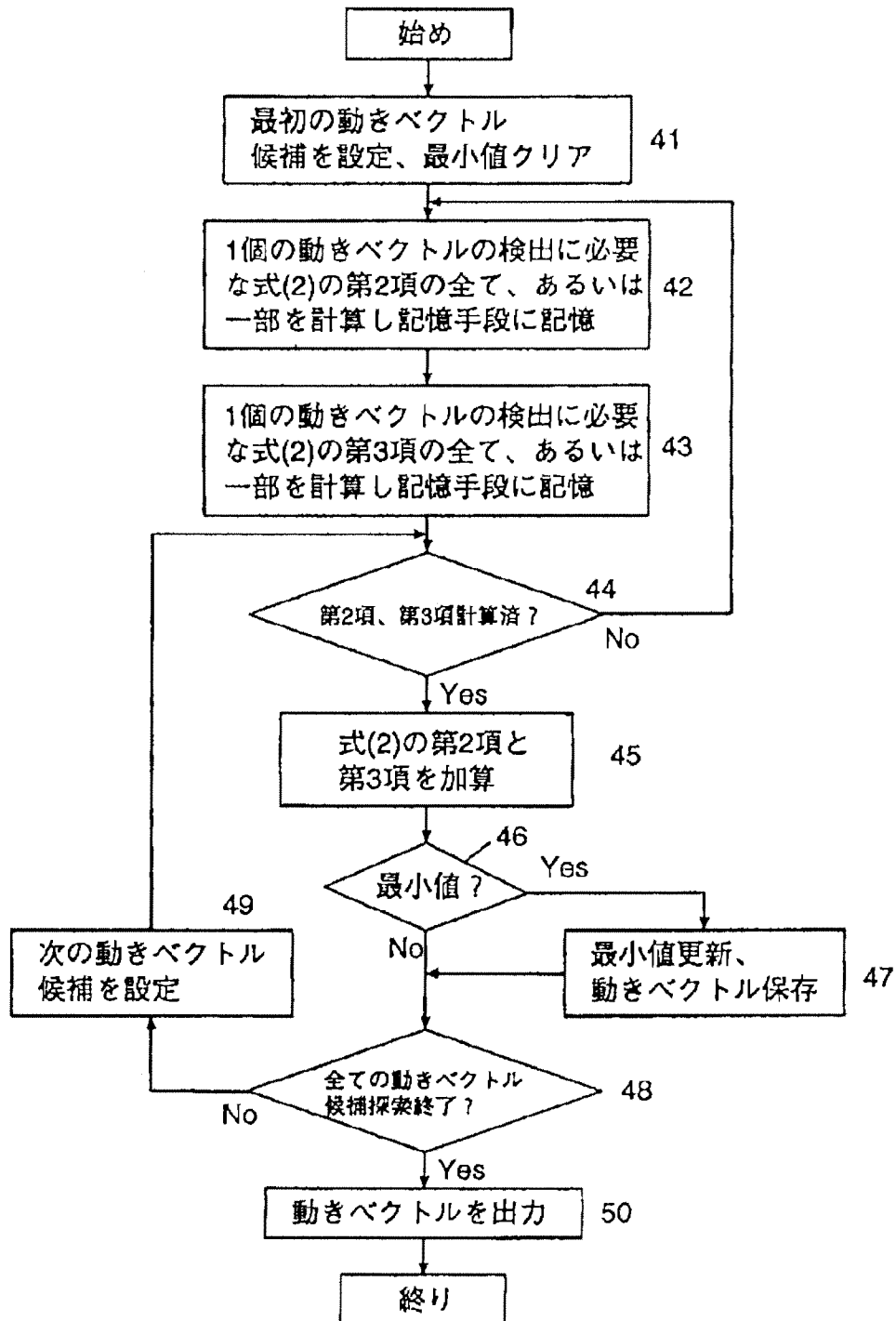
【図3】



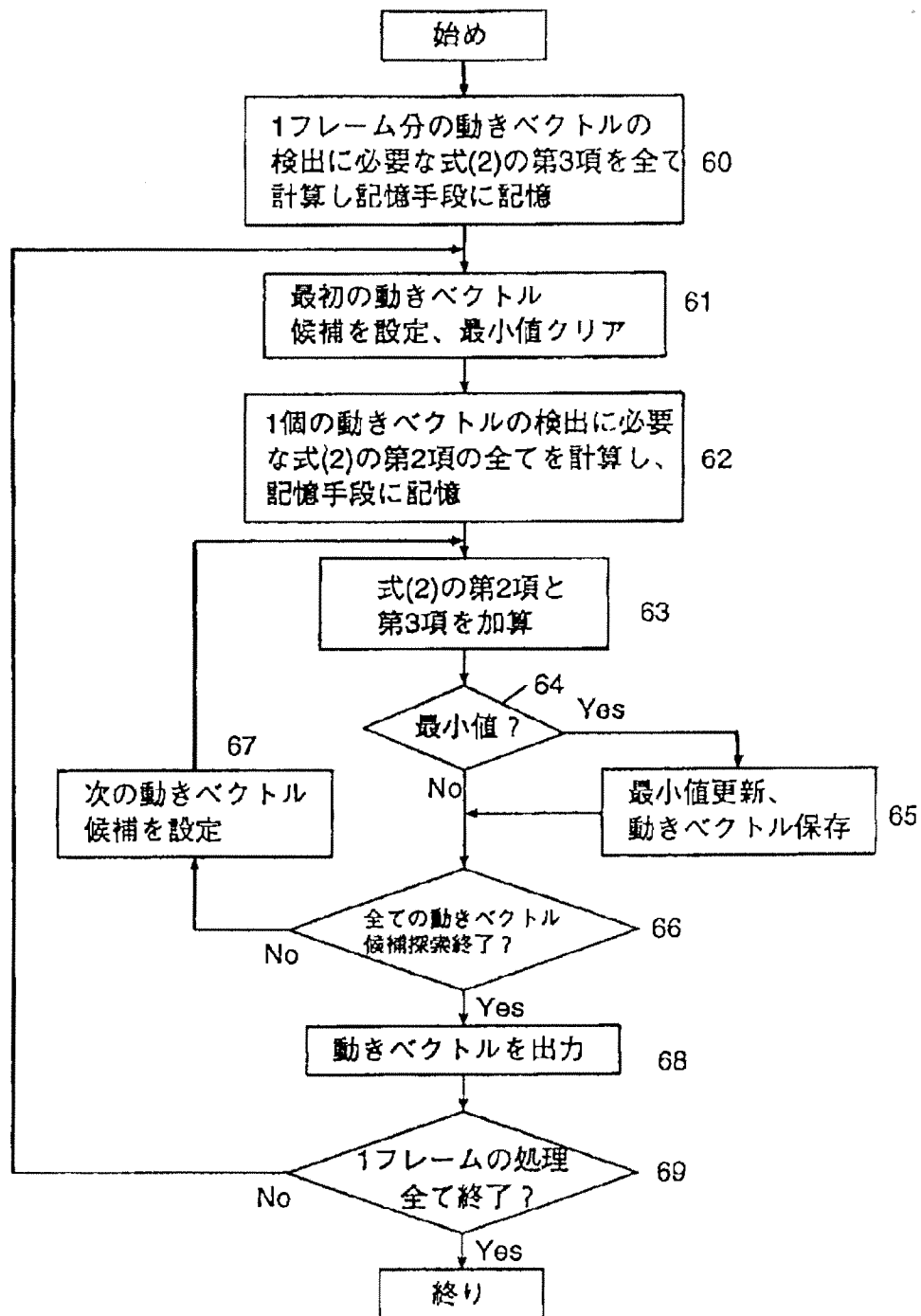
【図15】



【図 4】



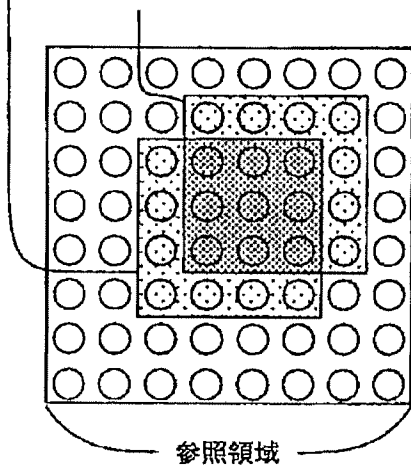
【図5】



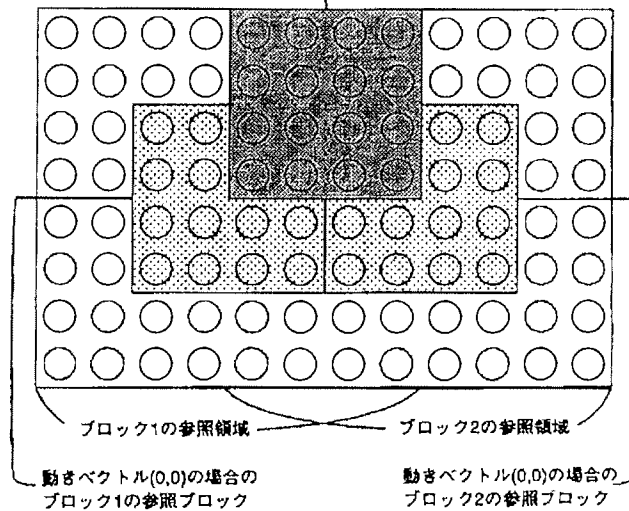
【図8】

動きベクトル(0,0)の場合の参照ブロック

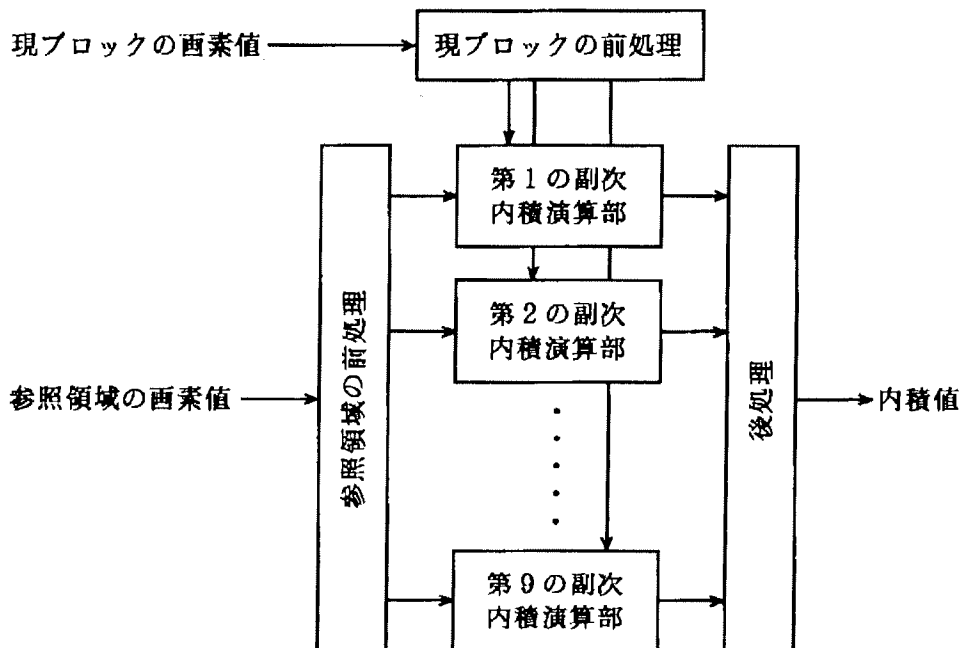
動きベクトル(1,-1)の場合の参照ブロック



【図9】

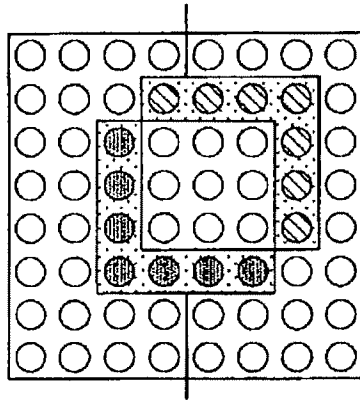
動きベクトル(2,-2)の場合の
ブロック1の参照ブロック動きベクトル(-2,-2)の場合の
ブロック2の参照ブロック

【図10】



【図 1 1】

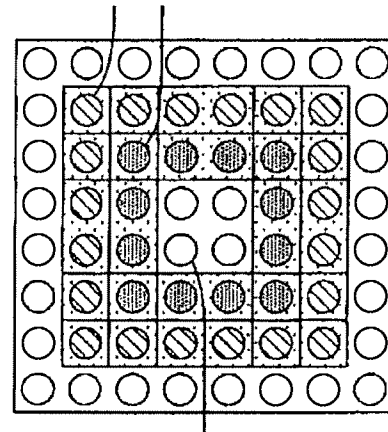
基準となる参照ブロックからみて
動きベクトルが(1,-1)の参照ブロック



基準となる参照ブロック

【図 1 2】

自乗値の計算が必要な画素



自乗値の計算が不要な画素

【図 1 3】

